Creating and Analysing Rauzy Graphs for DNA Sequences

by

David Robert Marcel Nahodil, BComp

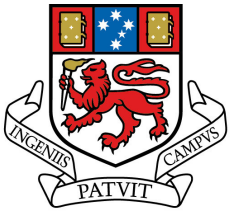
A dissertation submitted to the

School of Computing

In partial fulfilment of the requirements for the degree of

Bachelor of Computing with Honours

November 2005



UNIVERSITY
OF TASMANIA

Declaration

I, David Nahodil, declare that this work contains no material which has been accepted for the award of any other degree or diploma in any other tertiary institution, and to my knowledge and belief the thesis contains no material previously published or written by another person except where due reference is made within the text of the thesis.

_____

Abstract

Bioinformatics is a new and interesting field of research, and genuine discoveries are being made all the time. This project aims to investigate the application of a mathematical tool, called a Rauzy graph, to the problem of classifying DNA sequences. This project aims to learn more about the properties of DNA, and the nature of the graphs themselves.

The work for the project involves developing an algorithm for the computer to generate Rauzy graphs. Different sets of DNA sequences are used to build graphs from, and their properties analysed to investigate their behaviour in the biology domain.

The results of the work show that differences in DNA sequences do affect the structure of their Rauzy graphs and that, through this, the properties of the Rauzy graphs of DNA sequences could be used to classify those sequences.

Acknowledgments

First of all I would like to thank my supervisors, Arthur Sale and Andrei Kelarev for putting forward an interesting and challenging research project and for their ideas and support through the year. Arthur and Andrei were also responsible for the “Introduction to Bioinformatics” unit which ran this year. Through both my thesis project and the unit they have sparked an interest in the field of bioinformatics for me.

In the same vein I would also like to thank Bob Elliot from the School of Plant Science. Bob delivered the biology section of the bioinformatics unit. He did a great job of teaching biology to someone who last covered it in year 11 physical science and he is the master of asking challenging questions (and answering our questions with other questions).

My next thanks go to my family, Dad, Mum and Carla. Although you have all moved away and left me, you have always been, and continue to be, supportive. Thanks Carla for always telling me how great living in Melbourne is. Thanks to Mum and Dad letting me live in the house while you are away (I will vacuum soon, I promise).

Now to my wonderful girlfriend, Diana. Thankyou for being so supportive throughout the year, and all the years I have known you. Your advice for all things honours has been great, and you have always let me see things in perspective. I couldn't ask for a better girl to spend my (few) free hours with. Oh, and thanks for helping me decipher some of Bob's more detailed questions.

As for my non uni-type mates: Sam, Maz, Ben, Dane, Ollie and everyone from the Eastern Shore Rover Crew. Thanks guys for sticking by me when I needed

it and distracting me when I didn't. Thank you for pointing out that it is not cool to come home from Uni at 2:00am, as if I didn't already know that! Thanks to everyone from ESRC for giving me a real life outside Uni, I can't think of a crew I would rather be a part of.

Finally I need to thank all the guys from honours, this year wouldn't have been anything without you. I respect all the support and friendship you guys have given me throughout the year and I hope that I have done the same for you. I will look back at the time I spent with you more fondly than I could have hoped. Cheers.

Table of Contents

Tables of Figures, Algorithms and Equations

Table of Figures

Table of Algorithms

Table of Equations

1 Introduction

Since the discovery of the deoxyribonucleic acid (DNA) molecule structure, it has been one of the main focus areas within biology, as every living thing, from the smallest bacterium or virus, to the largest plants and animals has DNA. DNA defines how each organism will grow and function within its environment.

In recent years, one of the major new areas of research has been bioinformatics. Bioinformatics is the field of applying tools and techniques from mathematics and computer science, to research in biology. Bioinformatics is hoped to revolutionise the way in which biology is understood and studied.

From a biological point of view, to "classify" a DNA sequence would generally imply determining its lineage, family and species. From a mathematics and computer science point of view, to "classify" a DNA sequence would be to evaluate some or all of its properties (without interpreting its meaning) and categorise it with sequences which share similar properties.

This project focuses on using a special type of graph, known as a *factor graph* or *Rauzy graph*, as a tool for analysing and extracting properties of DNA sequences. Rauzy graphs have been used previously in studies involved in measuring complexity of sequences, often infinite sequences of numbers, and this work represents a major departure from previous applications of these graphs.

The aim of the project is to investigate the results of analysing Rauzy graphs which have been built for DNA sequences. Two things were hoped to be

achieved. Firstly, to investigate the properties of the graphs of DNA sequences and to make some decisions about whether they may be of use in classifying the sequences. Secondly, it will be a test of Rauzy graphs in a domain different from those it has been used in before, and it will be interesting to investigate how they behave.

This project involved developing an algorithm for the efficient generation of Rauzy graphs for given sequences, specifically DNA sequences. Subsequently graphs were generated for 45 sequences of DNA from chloroplast cells in different species of *Eucalyptus*. Data from the generated graphs was then collected, collated, and analysed to produce the results from which the conclusions were drawn.

This project provided data and results that not only supported the hypothesis, but also provided insight into the properties of Rauzy graphs when they were applied in this new field of bioinformatics.

2 Literature Review

2.1 Introduction

The classification of biological samples can be undertaken using various methods. Until a few decades ago, it was primarily anatomical homology (studying inherited physical features) and basic chemical analysis (Campbell et al. 1999). More recently, the primary methods have become those based on DNA sequencing techniques.

2.1.1 Background Biology

It has been known for a long time that DNA is the hereditary material through which all organisms inherit the traits of their ancestors. Since 1953, when the structure of DNA was discovered by James Watson and Francis Crick (Watson & Crick 1953), more and more interest has been invested in using DNA to study inheritance and aid in phylogenetic analysis (Campbell et al. 1999).

DNA is a double-stranded polymer molecule which has a 'backbone' of sugar-phosphate molecules, and one of four nitrogenous bases. These bases may be adenine, cytosine, guanine or thymine (represented as A, C, G and T respectively). A diagram of a DNA molecule can be seen in Figure 2.1.

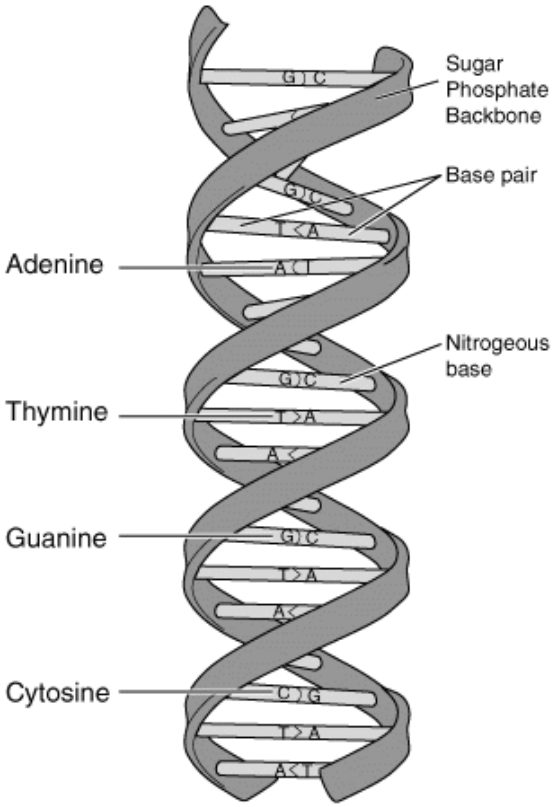


Figure 2.1 - The structure of a DNA molecule (courtesy (National Human Genome Research Institute).

The two strands of DNA form the well-known double-helix structure. These strands are complementary; each nucleotide base will only pair up with its complementary base, in accordance to the number of hydrogen bonds between them. A and T bases are bound together by 2 hydrogen bonds, and C and G are bound by 3 hydrogen bonds. Therefore, even though the DNA strands have different bases at the same position, they both carry the same genetic information (Campbell et al. 1999).

DNA, together with ribonucleic acid (RNA), forms a group classed as the nucleic acids. The nucleic acids are responsible for many of the complex processes that occur within living organisms. Most importantly, DNA is responsible for passing hereditary information through generations (Campbell et al. 1999). DNA is considered to be the 'code' that tells our cells how to function. Varying DNA sequences will result in different functionality and consequently different organisms (Campbell et al. 1999).

Phylogeny is the biological study of the evolution of species. A phylogenetic analysis attempts to learn more about the evolution of a particular species. More specifically "Phylogenetic analysis seeks to infer the evolutionary history that is most consistent with a set of observed data" (Li et al. 2000).

How DNA changes

Over time, the DNA of a species of organism can change. These changes can be of two different types, indels and substitutions. An indel (short for insertion/deletion) is an event where a nucleotide is inserted or removed from the DNA sequence. When comparing two sequences, it makes no difference if an indel event was an insertion or deletion (Campbell et al. 1999; D'Antonio 2003).

```
Sequence 1:- ACTTGATTC-TA
Sequence 2:- ACTT-ATTCTTA
                   ^    ^
```

Figure 2.2 - Two examples of indel events in a pair of sequences. In these examples a '-' represents a gap in the sequence.

Two examples of indel events are shown in Figure 2.2. The event at position 5 in the sequence could be considered an insertion in sequence 1 or a deletion in sequence 2. Conversely the event at position 10 could be considered an insertion in sequence 2 or a deletion in sequence 1.

```
Sequence 3:- ATTCGAGTTA
Sequence 4:- ATTTGAGATA
                  ^
```

Figure 2.3 - An example of a substitution mutation in a pair of sequences.

In sequences 3 and 4 (Figure 2.3), a substitution event at position 4 in the sequence can be seen. Once again, it is not significant to consider which sequence the change occurred in.

It is because of these mutations that organisms evolve over time, and it is these mutations that can be easily monitored by biologists which provide vital information about how a species has evolved.

Genes

It is interesting to note that most people, even those with no knowledge of biology, have heard of genes, even though there is no perfect definition of what one is. Generally, a gene is defined as a specific sequence of DNA or RNA that is responsible for encoding the amino acid sequences which produce specific proteins (Campbell et al. 1999).

The code contained within a gene is used to create messenger RNA (mRNA) through a process called transcription. mRNA is subsequently 'read' by ribosomes which produce proteins accordingly, which is called translation. Proteins are the workhorse of the cell. They facilitate the chemical reactions and are catalysts for the processes which occur within and between cells. It is through the processes of transcription and translation that changes in a DNA sequence affect the proteins produced by the cell, and consequently the organism it belongs to (Campbell et al. 1999).

Evolutionary Trees

In most cases, the aim of sequencing DNA to classify organisms is to create a phylogenetic tree. Generally a DNA sequence representing a specific gene (or genes) is analysed, and any sequences that are similar between samples can be used to try and determine an evolutionary path, so that classifications of that specific organism can be made. David Mount (2001) describes an evolutionary (phylogenetic) tree as "… a two-dimensional graph showing evolutionary relationships among organisms, or in the case of sequences, in certain genes from separate organisms". An example of an evolutionary tree can be seen in Figure 2.4. The species' names can be seen down the right hand side. The lines show the relationship between the species and the numbers represent an estimated "distance" between the species.

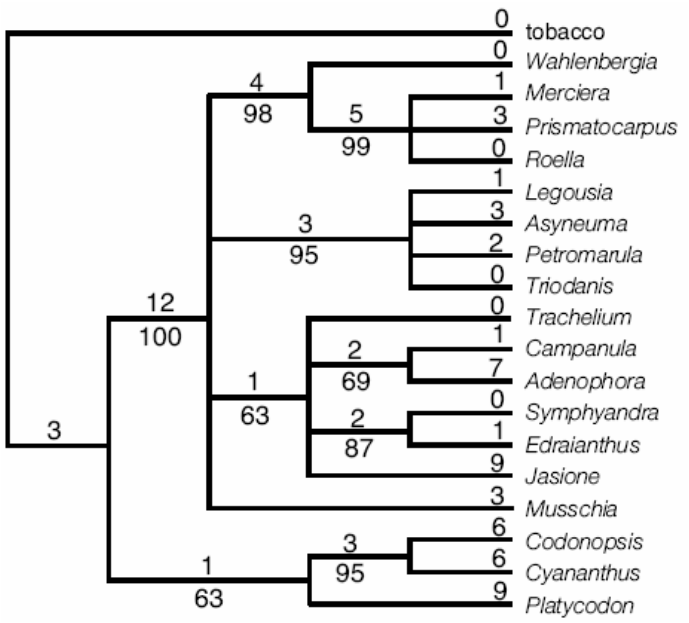


Figure 2.4 - An example of a phylogenetic tree (Cosner, Raubeson & Jansen 2004).

2.2 Literature

The problem of classifying organisms and creating phylogenetic trees has been rectified for some time. For creating phylogenetic trees, methods using parsimony, likelihood and distance metrics are almost 40 years old (Sanderson & Shaffer 2002) and are all still in wide use.

The most widely used methods have been those based on maximum parsimony, but these are best suited to sequences with strong similarity. Sequences which have weak, but still recognisable, similarities are better analysed using distance or maximum likelihood methods (Li et al. 2000; Mount 2001). These methods are discussed further shortly.

2.2.1 Current Techniques

There are different techniques for classifying DNA sequences. All of these methods are based on some knowledge of the domain; they have been developed explicitly to take known relationships and popular theories from biology into account.

Evolutionary Trees

The techniques of phylogenetic tree construction described here each fall into either of two categories:

- Exhaustive search
- Stepwise clustering

An exhaustive search technique is one where it examines a large number, or all, of the possible trees, and chooses one it considers 'the best' based on some chosen criteria. Maximum parsimony and maximum likelihood methods are both examples of exhaustive search methods and are described below (Saitou & Imanishi 1989). In these cases the names suggest which criteria the trees are selected on.

Stepwise methods are beneficial because they have lower memory requirements than exhaustive methods and can therefore be performed faster on larger collections of sequences. Nearest neighbour and other distance methods are stepwise methods and are described below (Saitou & Imanishi 1989).

Maximum parsimony / Minimum evolution

The maximum parsimony (or minimum evolution as it is also known) is based on the idea that the most simple (fewest evolutionary changes) tree that explains how the species evolved is the most likely. According to Mount (2001), the maximum parsimony method "predicts the evolutionary tree (or trees) that minimizes the number of steps required to generate the observed variation in the sequences".

The maximum parsimony method is guaranteed to find the tree of best fit, as it samples the entire search space. However, this is a slow process and is not suitable for large groups of sequences. Given the algorithm, the sequences must be aligned using a MSA algorithm before they can be used to build a tree. This further increases the complexity of this method (Mount 2001).

Distance Method

The distance method is the simplest approach to building a tree. It looks at all of the sequences to be aligned in pairs, and between each of the pairs the number of differences is calculated. The pairs with the fewest differences are deemed 'close'. Building the tree is just a matter of putting the closest sequences next to each other in the tree, then the next closest, and so on, until all of the sequences are in the tree (Mount 2001).

Neighbour joining

The neighbour joining method created by Saitou and Nei (1987) attempts to replicate the results of maximum parsimony methods (one unique tree for any set of sequences) but with the efficiency of a stepwise method. The neighbour joining method is reliable at constructing the correct tree where the rate of evolution vary between the sequences (meaning the branch lengths vary) (Mount 2001).

Maximum likelihood

The maximum likelihood approach is similar to the maximum parsimony method in three ways. First of all it is an exhaustive search, considering all of the possible trees. Secondly it is based on sequences which have been aligned using a MSA algorithm. Finally, this approach attempts to find the trees with the shortest evolutionary path, based on the assumption that they will be more likely to occur. The major difference is that the maximum likelihood approach is able to evaluate trees in which each branch evolves at a different rate (Mount 2001).

2.2.2 New Technique

Some of the early work on representing words with graph structures was done by N.G de Bruijn (de Bruijn 1946). One of the techniques he proposed was that of a De Bruijn graph. Later work in the field produced an idea known as a Rauzy (or factor) graph. Experimentation with these ideas has stayed mainly within the mathematics field but they pose interesting research opportunities in other fields, including biology (Berstel 2002; Jaeger et al. 2003).

A word, in the mathematical sense, is any sequence of symbols from a set, known as an alphabet. A word may be finite or infinite in length (Lothaire 2002).

De Bruijn graphs

A de Bruijn graph of order n is a labelled (directed) graph where the set of vertices (V) is every unique combination of elements of alphabet A of length n. The set of edges is (as given in by Lothaire (2002)):

$$E = \{(bs, a, sa) \mid a, b \in A, s\, A^{n-1}\}$$

Equation 2.1 - The set of edges in a de Bruijn graph.

This means that an edge s is an element of alphabet A going from a to b and s is a word of length n-1. As an example, Figure 2.5 is the de Bruijn graph over alphabet {a, b} of the order 2 (Lothaire 2002). The graph contains all words of length 2 that can be formed using the alphabet and the paths to get from one word to another.

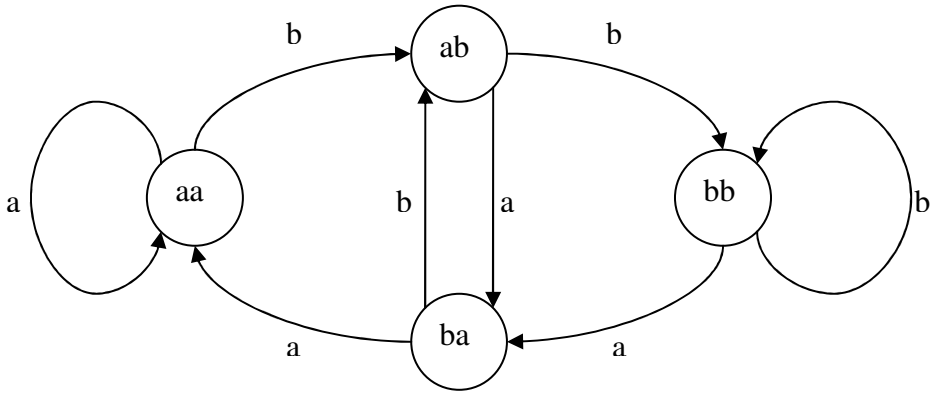


Figure 2.5 - The de Bruijn graph of order 2 of the alphabet {a, b}. Adapted from (Lothaire 2002).

Rauzy graphs

A useful tool when looking at the properties of a given word is an adaptation of de Bruijn's original work by Gérard Rauzy known as a Rauzy graph. These graphs are built in a similar way to de Bruijn graphs but rather than each vertex

being a word of length *n* from the alphabet given, each vertex is a sub-word (or factor) of a given word. For example: the Rauzy graphs of the word 'bananas' of orders 2 and 3 can be seen in Figure 2.6.

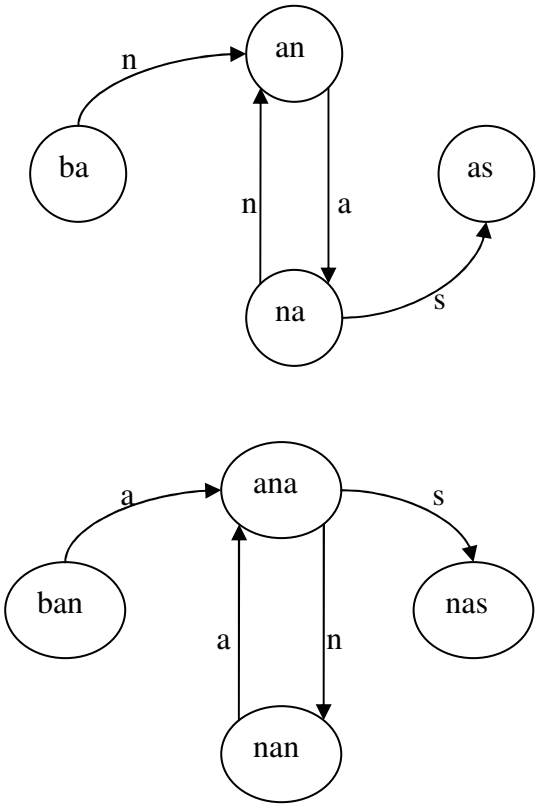


Figure 2.6 - The Rauzy graphs of orders 2 and 3 of the word 'bananas'.

Rauzy graphs can also be built for finite or infinite words. However the main restriction is that they must occur over a finite alphabet (Cassaigne 1995; Frid 2001).

Current studies

Rauzy graphs were first introduced a little over 20 years ago. Since then most of the work undertaken has been on infinite words with low complexity. It has been found that Rauzy graphs provide a good way to analyse these kinds of words (including Sturmian words and DOL sequences)(Berstel 2002; Frid 2001).

These analyses are all of a mathematical nature, and although some interesting properties have been found using these techniques, they have not been employed any wider than that.

Application in Bioinformatics

This project aims to apply the use of Rauzy graphs to the field of Biology. Rauzy graphs will be built off DNA sequences and analysed with the aim of determining how useful the properties of the graphs generated are with respect to classifying the DNA sequences.

Why Rauzy graphs?

There are many reasons why it may be of interest to analyse non-mathematical words using these kinds of techniques. A few reasons relevant to this project are discussed below.

Firstly, the application of a Rauzy graph-based analysis to DNA sequences will be an interesting investigation. This project provides an opportunity for observation of how Rauzy graphs behave when applied to sequences from biology, a field that is very different from those in which Rauzy graphs have previously been employed.

Secondly, it is possible that attempting analysis in this way – which is a completely different approach to the current methods – may provide new insight into the test sequences, or DNA in general. There are examples of machine-learning methods having provided new evidence that contradicts a long-standing theory, including within the field of bioinformatics (Towell & Shavlik 1994).

Finally, and perhaps most importantly, the problem of classifying DNA is one of analysing a finite word over a finite alphabet (A, C, G and T) making it a perfect domain to test these techniques. Not only can graphs be easily built representing DNA sequences, but also the factors of a DNA sequence (the nucleotides) have such an importance to the sequence itself. It doesn't take much to realise that the sub-words of length 3 of a DNA sequence are in fact

the codons that determine precisely which amino acids are produced and subsequently which proteins are present in the cell. (Campbell et al. 1999)

Properties of Rauzy graphs

The results from tests performed depend greatly on how the results are measured, and how the different tests are conducted. Given the somewhat limited analysis of Rauzy graphs in the past, it is not clear how they are best analysed. Some of the choices about how they should be constructed are also unclear.

When building the Rauzy graphs, a range of values should be chosen for the length of the factors. The length of the factors will affect the vertices in the graph and its structure. Initially graphs of orders 2 through 10 will be generated. Based on the results from those graphs it may be necessary to generate more graphs.

Another problem to consider is, if the sequences are aligned beforehand, how any gaps in the sequences are treated. Possible options are to treat a gap as any of the four possible bases, or as none of the other bases. The first of those two options would treat the sequence '`AACT`' as the same sub-word as '`AA-T`', the second option would treat them as different subwords of the sequence.

One of the properties of vertices in directed graphs (including Rauzy or de Bruijn graphs) is their 'type'. A vertex's 'type' describes how many edges lead into (a vertex's in-degree) and out of it (the out-degree). There are 4 possible types of vertex, examples can be seen in Figure 2.7 (Berstel 2002; Frid 2001; Lothaire 2002). Vertex A is known as 'ordinary' because both its in- and out-degrees are less than (or equal to) one. Vertex B is right-special because it has an in-degree of one and an out-degree less than (or equal to) one. Vertex C is left-special because it has an in-degree greater than one and an out-degree of one. Vertex D is bi-special because both its in- and out- degrees are greater than one.

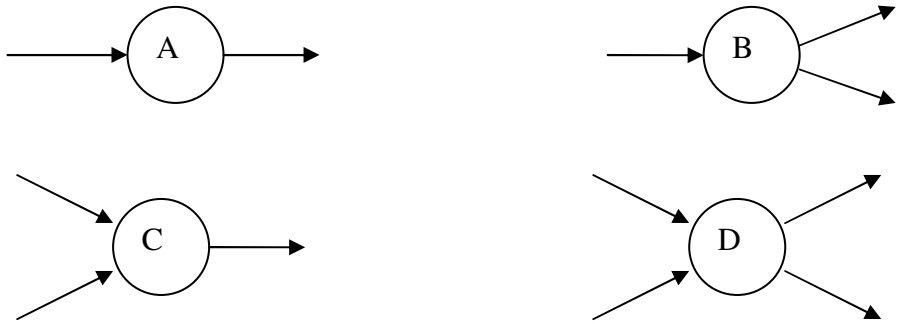


Figure 2.7 - Four examples of the possible 'type' of vertices.

2.3 Summary

There are problems with classifying samples based on their DNA sequences. However, there are a few possible ways in which these problems can be overcome. The various procedures for constructing phylogenetic trees are based on differing knowledge and assumptions based on the sequences. Although these techniques are quite proficient in producing trees, they all approach the problem from a biological perspective.

By attempting to classify DNA sequences from a completely different angle, in this case using a mathematical-based analysis, it is possible that much knowledge can be gained. Not only may a new technique or idea result, but it is also possible that a deeper understanding into the nature of DNA sequences could be achieved.

3 Methods

3.1 Aims

After considering the aims of the project the hypothesis was set down:
It is possible to classify DNA sequences by computing their Rauzy graphs and evaluating the graph's properties.

To test the hypothesis, it was fundamental to be able to generate and analyse the Rauzy graphs of the provided sequences. The aim for the project was to develop an algorithm for creating these graphs, and then to implement it in software in a way which allows properties of the graphs to be collected.

3.2 Considerations

3.2.1 Platform / Existing Software

The platform choice was fairly open, as the project had no requirements to work with any other software or system. There is no existing software directly relevant to this project, so integration was not a factor when considering a platform.

There were two general options in terms of development, either write a stand-alone program, or implement the algorithm in an existing software package. Both of these options were investigated. A few languages were considered (C, C++, Java) and also a software environment called "R" (R Development Core Team 2005). R is not only a development environment, but also a statistical analysis tool.

R was originally chosen because of the combination of development and analysis in one package. R also has support for additional packages, including "Bioconductor", (Gentleman et al. 2004) which includes tools for bioinformatics research. Further investigation later revealed that the tools in the Bioconductor package were inappropriate for this project, as this was not a traditional bioinformatics problem. From that point, development of the algorithm was continued in Java.

Existing code for reading and writing files was modified to suit the implementation. The original code was written by Phillip Uren from the School of Computing at the University of Tasmania (2005).

3.2.2 File Formats/Data Types

It was decided that the program should accept DNA sequences in the standard FASTA file format. The definition of the FASTA file format can be viewed at http://www.ncbi.nlm.nih.gov/blast/fasta.shtml (National Center for Biotechnology Information). Using the current setup, the program accepts each sequence as a separate file, with the first line leading with a '>' and containing the sequence name and optional comments (the "defline"). From the second line is the DNA sequence itself, using 'A', 'C', 'G' and 'T' characters.

It was important to format the output in a way that conveyed all the information about the graph, for testing and debugging as well as for collecting results. The most obvious way to display a graph is as an image showing the layout of the graph with the nodes and edges labelled. Although this method of presentation is suitable for testing and debugging purposes, it is not much good for analysing the results. When displaying the Rauzy graphs pictorially, the layout of the graph (which is determined by the layout algorithm) could be deceptive and suggest results which are purely artefacts of the layout algorithm. This is due to the algorithm choosing where nodes are placed and arranged, and different layouts can hide or accentuate different properties.

3.2.3 Time Constraints and Efficiency

Because of the nature of the project, there were no specific time constraints that the implementation was aiming to meet. The complexity for both building and printing the graphs have been kept as low as possible without obfuscating the code. In the final program much of the execution time is due to file operations and any output to the console that the operator chooses to display.

Algorithms to perform certain operations on list and graph structures are notorious for having poor time complexity, especially when used on large datasets. For this reason, the program had to be constructed carefully to make the best (and minimal) use of list operations. To avoid the problems which graph-traversal algorithms bring (where to start, how to handle loops, etc.), there are no actual graph data-structures in the program (multiple incoming and outgoing edges), only lists.

3.3 Implementation

3.3.1 Data Structures

To understand the representation of the Rauzy graphs used by the program a short description of the various data structures used within the program is given below.

Nodes

A Node object stores the information about one vertex in a Rauzy graph. Each Node stores only the sequence it represents (a portion of the original sequence whose length is the same as the order of the graph) and an ID number.

Edges

An instance of the Edge class represents one edge from the graph. Each instance of the Edge class has:

- A reference to the Node the Edge is from
- A reference to the Node the Edge is to

- The label of the Edge
- An ID number

NodeList

A NodeList object is used to create a traversable list of Node objects. Each NodeList object stores an ID number, a reference to its data (a Node object), and a reference to another NodeList object.

EdgeList

An EdgeList object is similar to a NodeList except, the data it references are objects of the Edge class, and it links to other EdgeList objects.

RauzyGraph

One of the main decisions was representation of the data structures within the program, specifically how to represent a Rauzy graph. The obvious option was to directly represent the Rauzy graph and its structure as a directed graph structure made up of linked Nodes. However, due to the added complexity of operations on graph data-structures over lists, this idea was not implemented.

With time and programming complexities in mind, the best option was to store a list of Edges (representing the edges between vertexes of the graph). Each of the Edges comprises of two Node objects (representing nodes from the graph), and a label for the Edge which is the letter that is suffixed onto the first node to make the second node (conforming to the definition of a Rauzy graph).

In this way the internal representation of the Rauzy graph is the complete list of all Edges that appear within that specific graph at that specific order (sub word length).

An object of the RauzyGraph class contains the following information about the graph it represents:

- The defline of the sequence it was built from
- The sequence itself
- The alphabet size of the sequence (the default is 4, for DNA sequences)

- The order (sub-word size) of the graph
- A list of the Edges that comprise the graph
- A list of all Nodes in the graph
- The adjacency matrix representation of the graph (if calculated)
- An ID number.

3.3.2 Algorithm

The basic outline of the algorithm can be seen below (Algorithm 3.1) along with more detailed descriptions of each step. The algorithm directly below is implemented in the *RauzyGraph* class (specifically the *build()* method).

```
If order of the graph > length of sequence
   Print error message
Else
     For each substring of length order
           If it exists in the list of Nodes
              Add a reference to the existing Node to the list
           Else
              Add new Node to the list of Nodes

     For each element in the list of Nodes
           If next Node exists
                Create an Edge from this Node to the next Node
                Add the new Edge to the list of Edges

     Mark the graph as having been built
```

Algorithm 3.1 - The algorithm to create Rauzy graphs from a sequence to a specific sub-word length (order).

A common representation of graphs (both directed and undirected) (Preiss 1997) is the adjacency matrix. This is a 2-dimensional array with each row and column representing a specific vertex. Each cell has a value of either ‘0’ or ‘1’. If there is a ‘1’ in a cell it means there is an edge from the vertex represented by the row to the vertex represented by the column. A ‘0’ in a cell means there is no edge from the vertex represented by the row to the vertex represented by the column. The adjacency matrix for an undirected graph will be symmetrical across the top-left to bottom-right diagonal.

For the sake of portability, this program contains code to create adjacency matrices from the Rauzy graphs. Doing this allows the graphs to be used with a wider variety of programs and algorithms. Algorithm 3.2 shows the algorithm used for creating an adjacency matrix of a Rauzy graph. This algorithm will only work for Rauzy graphs using the same representation used in this project.

```
For each Node in the list of Nodes
     If ID number of Node < minID
          minID = ID number of Node
     If ID number of Node > maxID
          maxID = ID number of Node

Create a 2D array of size (maxID - minID) by (maxID - minID)

For each Edge in the list of Edges
     fromID = ID of from Node for this Edge
     toID = ID of to Node for this Edge
     Set value of cell [fromID-minID][toID-minID] to 1.
```

Algorithm 3.2 - The algorithm used for building an adjacency matrix of a Rauzy graph.

3.3.3 Output Style

There are 2 different output styles used in the final program. The first is a complete output of the graph, including all of the Nodes, all of the Edges and some statistics about the graph itself. It also includes the adjacency matrix representation of the graph. The second is a concise summary of all of the statistics about the graph. The concise style is used for comparing graphs side-by-side.

The output examples below come from the graph of the *Aromaphloia* sequence with a sub-word length of 2.

Complete output

This style of output allows all the information about a graph to be represented in an easy-to-read format. The output is made up of 5 sections. An example of the general information about the graph can be seen in Figure 3.1.

```
================================================
  Graph of >aromaphloia ID[3]
  16 verticies (100.0%); 62 edges.
  Sequence = AATTTATTTC... Length = 522.
================================================
```

Figure 3.1 - An example of the head of the output file.

Information about each Edge in the graph includes the ID number, sequence of the Nodes it is between, and the type of Edge it is. A portion of an Edge list can be seen in Figure 3.2.

```
  Edges:

   Edge[1188] [35] {AA} -T-> [36] {AT}
   Edge[1189] [36] {AT} -T-> [37] {TT}
   Edge[1190] [37] {TT} (T)  [37] {TT} x->x loop
   ...
```

Figure 3.2 - Part of a list of Edges. The Edge ID number can be seen as well as IDs and sequences for both the Nodes in the Edge.

Information about each Node in the graph, includes the ID and sequence for the Node, as well as the in- and out- degrees for the Node. An example of this section can be seen in Figure 3.3.

```
Nodes:
   Node[35] {AA} In-degree: 4; out-degree: 4.
   Node[36] {AT} In-degree: 4; out-degree: 4.
   Node[37] {TT} In-degree: 4; out-degree: 4.
   ...
```

Figure 3.3 - Part of a list of Nodes from the complete output style. In this case the in- and out- degrees of the Nodes are the same, but this is not always the case.

Figure 3.4 is an example of the summary section, which is comprised of values summarising all of the nodes in the graph.

```
In-degree. min:  3, max 4.
Out-degree. min: 3, max 4.
Average (mode) in-degree: 4, out-degree 4.
Average (mean) in-degree: 3.875, out-degree 3.875.
Number of right-special: 0.0, number of left-special 0.0.
Number of bi-special: 16.0, number of ordinary 0.0.
Right-special (%): 0.0, left-special (%) 0.0.
Bi-special (%): 100.0, ordinary (%) 0.0.
```

Figure 3.4 - A summary section for a graph from the complete output style.

Finally, the adjacency matrix of the graph is shown. The rows of the matrix are labelled by the ID number of the Node they represent. An example can be seen below (Figure 3.5).

```
Adjacency Matrix:
 [35] 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0
 [36] 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0
 [37] 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0
 [38] 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0
 [39] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
 [40] 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0
 [41] 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0
 [42] 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0
 [43] 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0
 [44] 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0
 [45] 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0
 [46] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
 [47] 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0
 [48] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
 [49] 0 0 0 0 0 1 0 0 0 0 0 0 1 0 1 1
 [50] 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0
```

Figure 3.5 - An example of an adjacency matrix from the compete output.

Concise Output

The concise output is used for summarising the graphs of all sequences generated per order. It is comprised of many values each separated by a comma. The values are: the number of nodes in the graph; the number of edges in the graph; the number of all possible nodes (based on alphabet size and order) that occur in the graph (as a %); the number of nodes with an edge to themselves; the number of nodes with an edge to a second node with an edge to the first node; the minimum in-degree for the graph; the maximum in-degree for the graph; the minimum out-degree for the graph; the maximum out-degree for the graph; the most common in-degree for the graph; the most common out-

degree for the graph, the average in-degree for the graph; the average out-degree for the graph; the number of right-special nodes; the number of left-special nodes; the number of bi-special nodes; the number of ordinary nodes; the number of nodes that are right-special (as a %); the number of nodes that are left-special (as a %); the number of nodes that are bi-special (as a %) and the number of ordinary nodes (as a %). An example can be seen in Figure 3.6.

```
>aromaphloia,16,62,100.0,4,6,3,4,3,4,4,4,3.875,3.875,0.0,0.0,1
6.0,0.0,0.0,0.0,100.0,0.0
```

Figure 3.6 - Example of concise output for a graph.

3.3.4 Extra tests

After the initial work of building the graphs was done and data analysis had started, some interesting results emerged. To further investigate these results some graphs of different datasets were built for comparison. The data used comprised of three DNA sequences from animals, and five randomly generated sequences. More information about these sequences is given below (page 24). The program and output for these datasets were the same as for the chloroplast DNA, and no adjustments were made to the algorithm.

4 Results and Discussion

4.1 Data Sets

To generate results, a set of DNA sequences was needed from which to build the Rauzy graphs. The set of DNA sequences used were chloroplast DNA provided by the School of Plant Science, University of Tasmania. The DNA sequences come from different species from the Eucalypt family with each sequence between 460 and 570 bases in length. The sequences are provided in Appendix A.

Given that all of the DNA sequences came from chloroplast cells it is expected that they will produce very similar results. The reason for this is that the cells all perform the same function within different species, the DNA sequences are very similar and so too should be the graphs. This is beneficial for gathering results as it should make any trends or patterns in the data easy to identify, as opposed to graphs of sequences that are very dissimilar.

For comparison purposes, three more DNA sequences were chosen from the National Center Biotechnology Information nucleotide database (http://www.ncbi.nlm.nih.gov/). Three DNA sequences from animals were deliberately chosen to be different from the plant DNA sequences, as animals do not have chloroplasts. These sequences can be seen in Appendix B. The sequences were trimmed (where needed) to be within the same range of sequence lengths as the Eucalypt sequences.

A dataset of five ‘DNA’ sequences was also generated to investigate the effect of random sequences. The sequences are a random length between 460 and 570 base pairs. Each base was chosen randomly and all bases had a probability of

0.25 of occurring at each position in the sequence. These sequences are available in Appendix C.

The chloroplast dataset was used for most the tests. When it seemed that analysing some different sequences might yield some interesting results the tests were run again with the animal sequences or the random sequences.

4.2 Compiling Summary Data

Graphs were computed for orders 2 – 10 for each sequence. The summary data for each graph was compiled into a spreadsheet, one for each sub-word length. The compiled data can be seen in Appendix D. This data was then used for analysis of the use of Rauzy graphs. All of the various properties of the graphs were analysed and compared. Below is a summary of the significant results and a brief discussion about what each result may indicate.

4.3 Results

4.3.1 General

The number of nodes and edges for the graphs of order 2 – 10 for the *Nortonii* and *Cordata* sequence can be seen in Figure 4.1 and Figure 4.2. The results for the *Nortonii* and *Cordata* sequences are indicative of the results for the other sequences. All sequences follow a similar trend with regards to the numbers of nodes and edges.

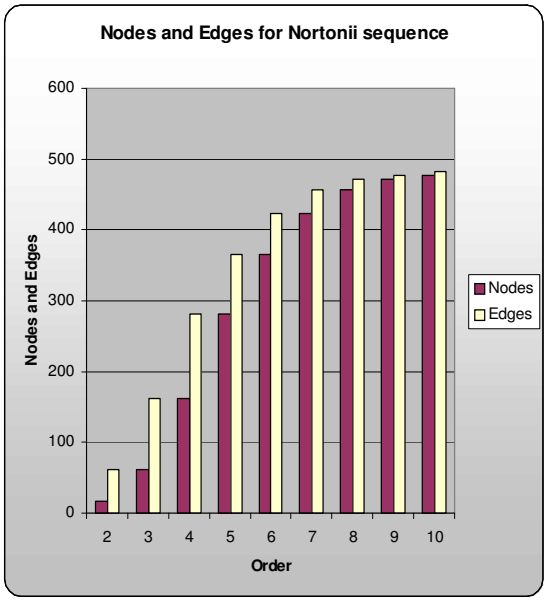


Figure 4.1 - Graph showing the numbers of nodes and edges in graphs of different sub-word lengths for the *Nortonii* sequence.

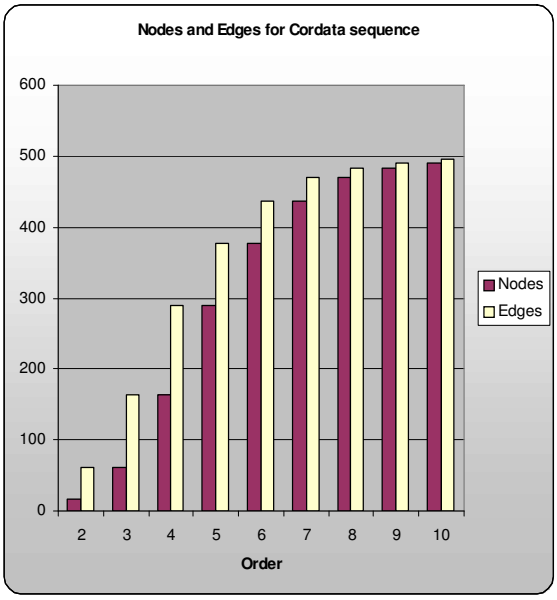


Figure 4.2 - Graph showing the numbers of nodes and edges in graphs of different sub-word lengths for the *Cordata* sequence.

It can also be seen that, for the *Cordata* and *Nortonii* sequences, the number of nodes in a graph of some order is the same as the number of edges there are in the graph of the same sequence with a sub-word length of *order-1*. This pattern holds for all the sequences tested (including the animal and random sequences) and for every sub-word length used.

The relationship between the edges of one graph and the nodes of the next graph is due to both the nodes, and edges, always being unique. This result also suggests that it is possible to adopt a dynamic-programming approach to the creation of Rauzy graphs, and, that rather than building each from scratch, a graph could be constructed from a smaller graph of the same sequence.

For all sequences, the size of the graphs taper off as the order of the graph gets higher. The size of the graphs with sub-word lengths 8, 9 and 10 grow much more slowly than when shorter sub-words are used. This pattern can be seen clearly in Figure 4.1 and Figure 4.2.

4.3.2 Graph Composition

Results were then collected by determining which types of nodes are present in each graph, and how many there are of each. Figure 4.3 shows the breakdown of each sequence, based on sub-word length. There are 45 sequences plotted in each column, however (as suspected) the graphs of all sequences have very similar values, and therefore appear as a small cluster.

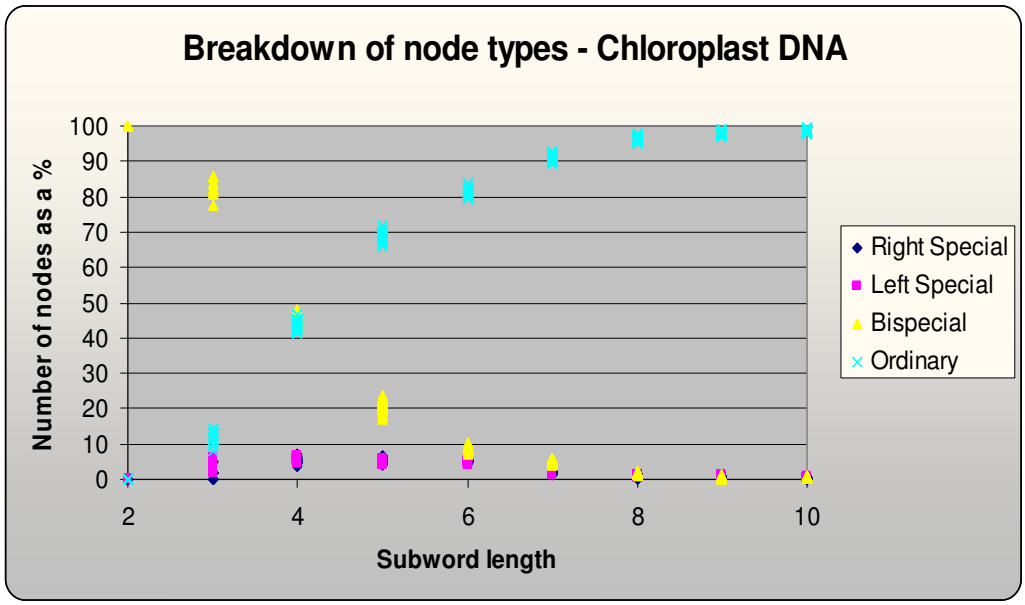


Figure 4.3 - Breakdown of node types in graphs of all sequences.

There are two important relationships demonstrated in Figure 4.3; the relationship between the proportion of bi-special and ordinary nodes, and that between the proportion of right- and left-special nodes. The relationship between the proportion of bi-special and ordinary nodes is approximately inverse. The right- and left-special nodes always have a very similar proportion, this is investigated further below.

It can be seen in Figure 4.3 that, for all graphs of all sequences the majority of nodes are either bi-special or ordinary. The graphs of low orders contain mostly bi-special nodes because these graphs are highly connected, every sub-word occurs many times in the sequence. In contrast the nodes of high orders contain mostly ordinary nodes. This is because it is less likely to have a long sequence occur twice (or more) in a DNA sequence. The middle orders (4, 5 and 6) display a greater balance between ordinary and bi-special nodes.

The relationship between bi-special and ordinary nodes indicates that as the order of the graph grows (that is the sub-words get longer) then the sub-words become more unique. For the higher orders (8, 9 and 10) this tends to a point where the Rauzy graph is merely a list of all the sub-words with an edge from one to the next. At the other end of the scale, the graphs converge at a point where all the nodes are bi-special. Again, this is due to the length of the sub-words. Graphs where the sub-words are only a few characters long have a very small number of possible nodes, and because those nodes occur frequently in the sequence, the all have many in and out edges. This is explained further, below.

A more detailed look at right- and left-special nodes shows the correlation in more detail. Figure 4.4, Figure 4.5 and Figure 4.6 show the proportion of right-special and left-special nodes for each sub-word length for all sequences. From order 4, these results show a tight cluster where the proportion of right-special nodes is very close to the proportion of left special nodes. For the graphs of the chloroplast DNA, the difference never exceeds 6.5%, and for the majority of cases, was less than 1 or 2%.

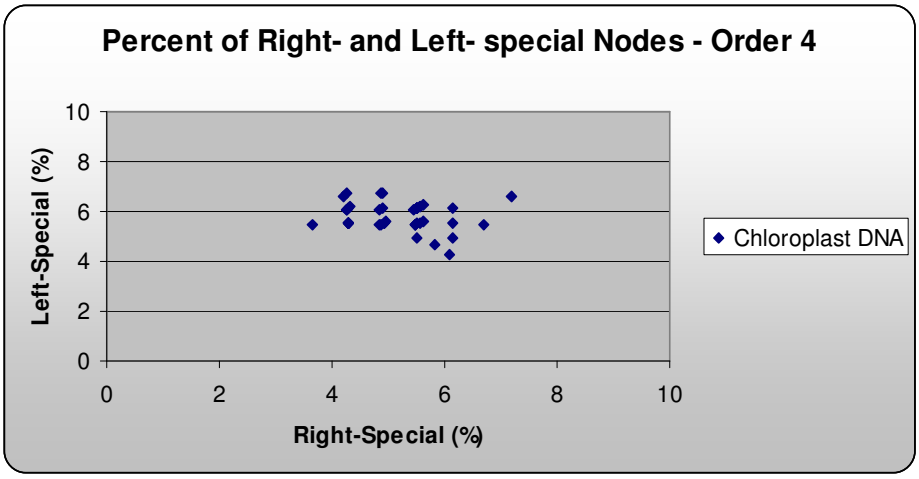


Figure 4.4 - Relationship between the numbers of right- and left- special nodes in each graph with subword length 4.

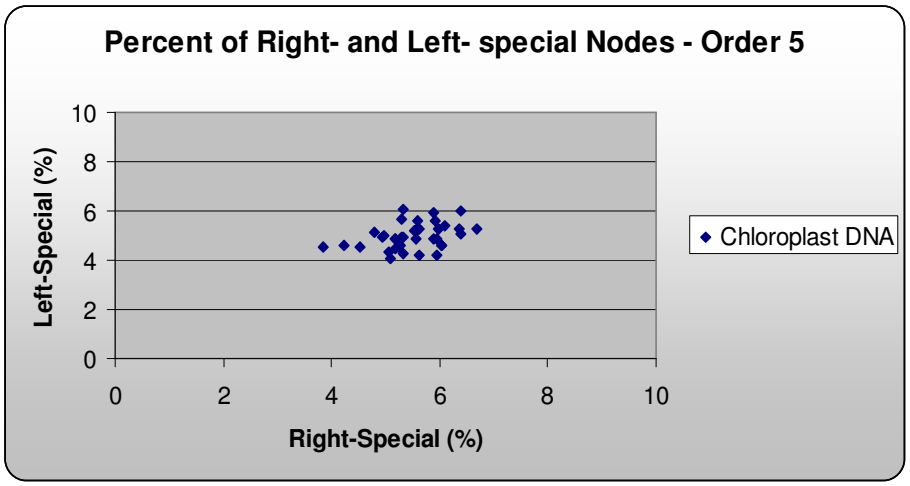


Figure 4.5 - Relationship between the numbers of right- and left- special nodes in each graph with sub-word length 5.

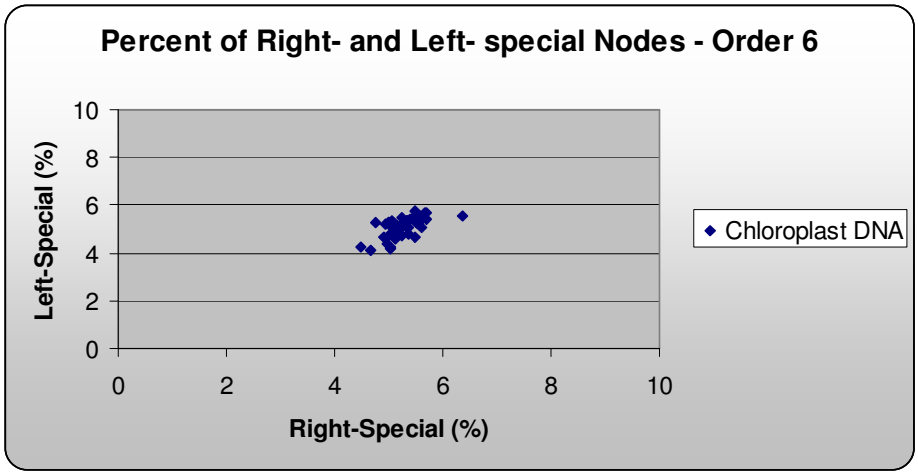


Figure 4.6 - Relationship between the numbers of right- and left- special nodes in each graph with sub-word length 6.

4.3.3 Graph Size

Figure 4.7, Figure 4.8 and Figure 4.9 show the relationship between the number of nodes in a graph with the number of edges in the graph. There are two series plotted, the Eucalypt sequences and the animal DNA sequences. It can be seen (especially in the graphs of orders 4 - 7) that the animal DNA form a cluster which is distinct from the chloroplast DNA. Both series settle down to form a trend where the higher the number of nodes in a graph, the higher the number of edges.

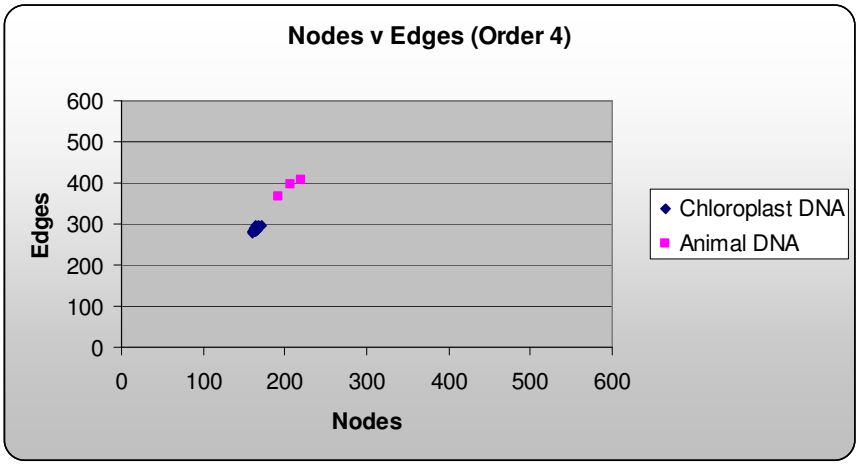


Figure 4.7 - Number of nodes versus edges for chloroplast and animal DNA.

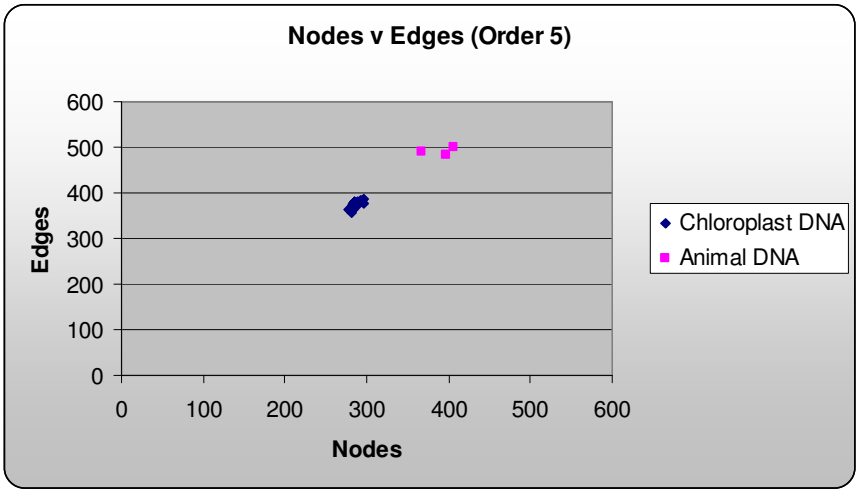


Figure 4.8 - Number of nodes versus edges for chloroplast and animal DNA.

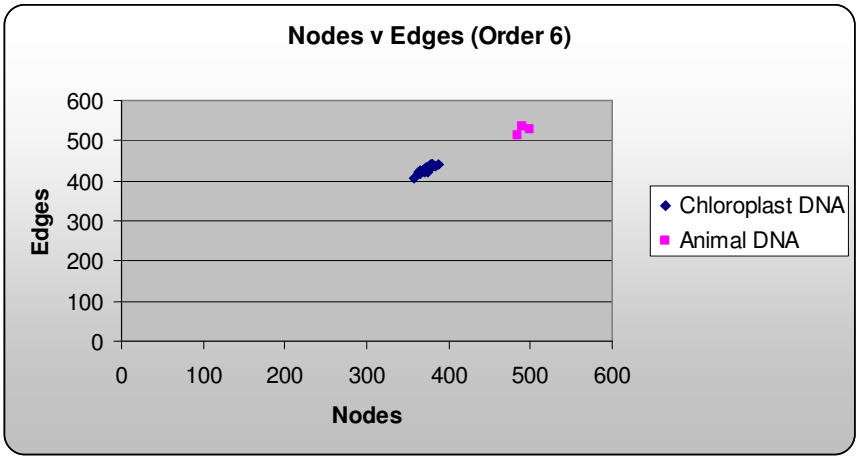


Figure 4.9 - Number of nodes versus edges for chloroplast and animal DNA.

The clustering shown in these graphs strongly suggests that the Rauzy graph of a sequence, specifically the number of nodes and edges, are representative of the sequences they are built from.

The chloroplast and animal datasets were then plotted against the random sequences. The result of this was that the random sequences grouped closer to the animal sequences than the chloroplast sequences; however they do not form the kind of close cluster that the other DNA sequences do. An example of this can be seen in Figure 4.10. The high variation, and random nature, of the sequences is reflected in their scattered position in this graph.

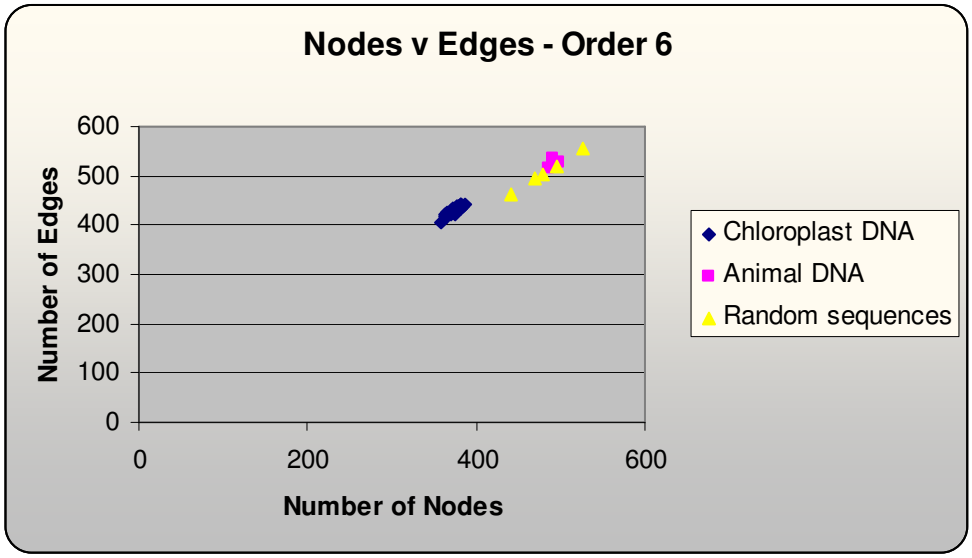


Figure 4.10 - Size of graphs in nodes and edges for all sequences (including random).

Figure 4.11, Figure 4.12 and Figure 4.13 show the number of nodes and edges in the graph of a sequence plotted against the length of the sequence. For graphs of lower orders there is little, or no, correlation between the length of the sequence and the number of nodes or edges. For graphs with orders above 6 the gap between the number of nodes and edges reduces, and a trend forms such that longer sequences have a higher number of nodes and edges, and shorter sequences have less nodes and edges.

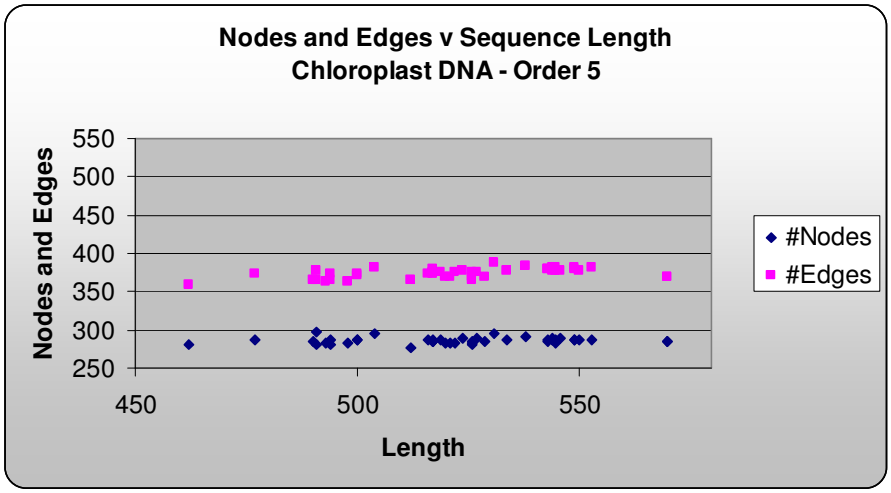


Figure 4.11 – Nodes and edges in each graph with sub-word length 5, based on sequence length.

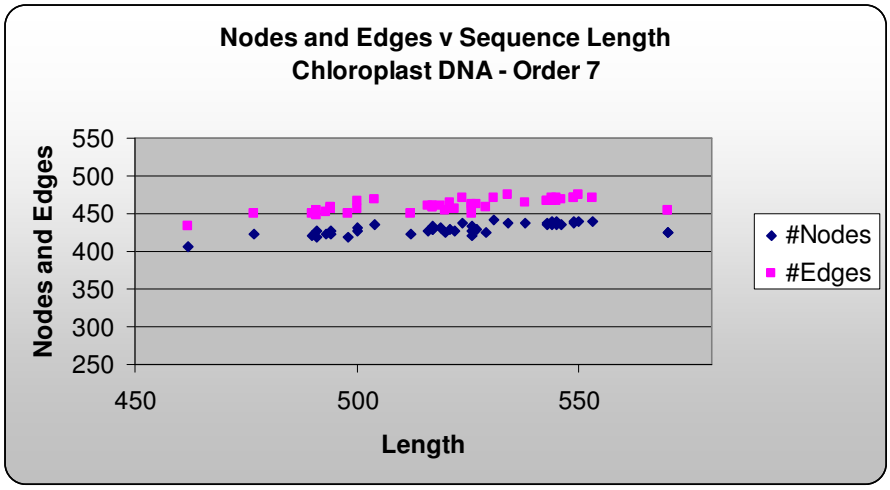


Figure 4.12 – Nodes and edges in each graph with sub-word length 7, based on sequence length.

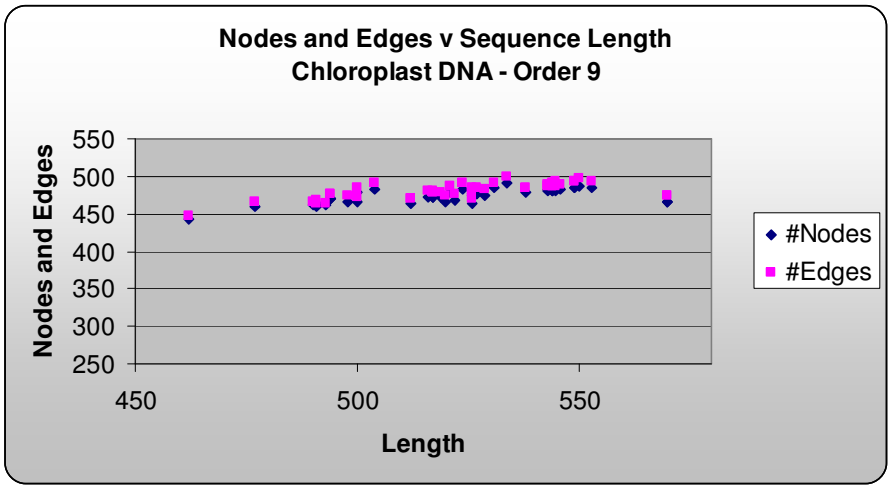


Figure 4.13 – Nodes and edges in each graph with sub-word length 9, based on sequence length.

The important point to note is that the size of the graphs is only slightly dependent on the length of the sequence. For graphs with low sub-word lengths, there is almost no correlation between the length of the sequence and the size of the graph (counting nodes or edges). For higher-order graphs there is a small trend that forms where graphs of longer sequences are larger on average, however, there is still deviation in the trend.

4.3.4 Graph Constraints

There are various ways of counting the number of nodes which are present in a graph. One way, is to express it as a percentage of the total possible nodes, based on order and alphabet size, as shown in Equation 4.1. Another way, is to express it as a percent of the possible nodes, based on order and sequence length, as seen in Equation 4.2.

$$n = s^a$$

Equation 4.1 - The maximum number of nodes in a graph, based on alphabet size.

The maximum number of nodes based on alphabet size, n, is given in Equation 4.1. Where s is the sub-word length (order of the graph) and a is the alphabet size. In the case of DNA sequences $a = 4$.

The maximum number of nodes based on sequence length, m, is given in Equation 4.2. Where l is the length of the original sequence and s is the sub-word length.

$$m = (l - s) + 1$$

Equation 4.2 - The maximum number of nodes in a graph, based on sequence length

The most nodes a graph can have is the minimum of the value of m and n as given in the equations above. It is easy to see that, no matter how long a sequence is, its order 2 graph can never have any more than 16 nodes ($2^4 = 16$). At the other end of the scale, the graphs of orders 9 and 10 are reaching the maximum amount of nodes, based on sequence length. When a graph has 100% of nodes based on sequence length, it means that the graph has become completely linear. We can see from this that the graphs of orders 8, 9 and 10 are approximately linear.

Figure 4.14 shows where the graphs of each order appear on the scales of the two metrics discussed.

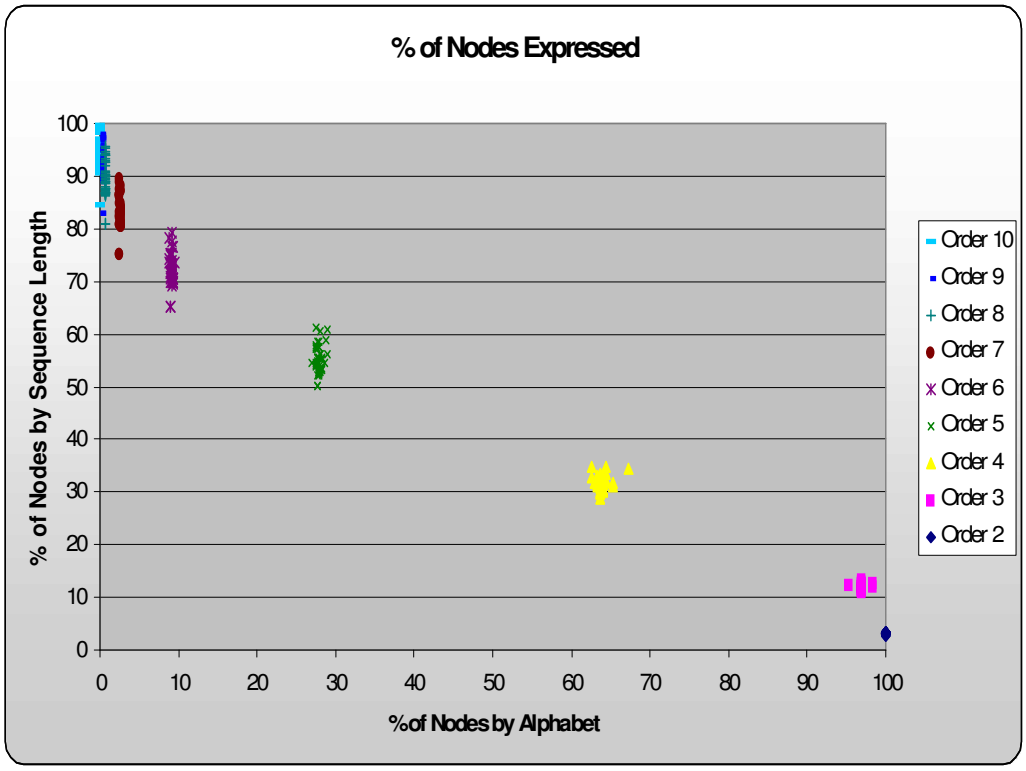


Figure 4.14 – The number of nodes in graphs of sequences of various orders expressed as % of alphabet size and sequence length.

This, once again, shows the results of graphs of sub-word lengths 4, 5 and 6 produced were within the middle of the range, and not restricted by either of the extremes.

5 Conclusions

Based on the results presented, it has been found that Rauzy graphs are representative of the DNA sequences they are built from, and differences in the sequences are reflected in the graphs. The size and other properties of the graphs are not determined by the length of the sequences, rather their contents.

When the graphs of the sequences were plotted with the number of nodes in the graph on one axis and the number of edges on the other, the DNA sequences from the chloroplast cells formed a tight cluster. When the graphs of the sequences of animal DNA were plotted on the same axis they formed a cluster away from the chloroplast DNA. Although, as only 3 animal DNA sequences were graphed, it was not clear how close a cluster would form if additional sequences were included. As a result, it was concluded that Rauzy graphs could potentially be used for classifying DNA sequences.

The graphs built with sub-word lengths of 4, 5 and 6 are the most sensitive to changes in the DNA sequences. In this range the differences between the graphs were the most pronounced, and the graphs were not limited by factors like sequence length of alphabet size. This is based on the results from DNA sequences that were between 460 and 570 base-pairs in length.

5.1 Secondary conclusions

When graphs of sequences were built with sub-word lengths of 2 and 3, the graphs did not convey any useful information about the sequence. The graphs of sub-word length 2 were almost all identical and there were still very few differences with a sub-word size of 3. It appears the problem stems from a limitation of the alphabet size.

Building graphs for sequences with a length of 460 - 570 base pairs also encounters problems when the sub-word length grows above 8. In this range, the number of unique nodes in the sequence is almost the same as the number of nodes in the graph itself. The graphs with high sub-word lengths have very few edges, and are more like a long list of nodes than a connected graph. For this reason, further work in this area need not build graphs with sub-word lengths greater than 8 (for sequences of this length), because they do not exhibit the features that a more connected graph should have.

In the graphs of the chloroplast DNA sequences, the percent of ordinary and bi-special nodes was approximately inverse. With every sequence having 100% of nodes being bi-special, on graphs of sub-word length 2, and all graphs having well over 90% of ordinary nodes when the sub-word length was over 8. For most graphs a small proportion of the nodes were right- and left-special, never more than a few percent.

6 Further work

The results of this project suggest further work on this problem is warranted. There are a few areas where this work could focus:

- Investigating the mechanisms behind the clustering
- Examining deeper properties of the graphs
- Other biological applications
- Modifying / Extending the Rauzy graph algorithm
- Developing classification system using Rauzy graphs

6.1 Clustering

More work is required to determine the factors which cause clustering and investigate cases where clustering may not occur. It is necessary to apply the Rauzy graph technique on more varied DNA sequences. Although some of the results showed clusters forming in the plant DNA and those sequences from animals, it is not clear why they formed these clusters. It may be a result of their animal or plant source, but it could be other factors. The different clustering could be due to the region of the cell the DNA came from, whether or not it is a coding region and which part of a protein it codes for. There are many properties which define DNA that are not simply whether it came from a plant or animal.

6.2 Graph Properties

Through the collected data, the properties of the graphs that produced the most meaningful results were those related to the numbers of nodes and edges. These provided a measure of the size of the graph, and a measure of the uniqueness of the sub-words in the sequence. Some of the other properties of

the graphs were not analysed fully. Further work on this limitation might involve investigating:

- Larger loop structures in the graphs
- The importance of which nodes are right- of left- special
- The importance of wholly connected nodes. That is node with an in- and out- degree of 4.

Investigation into other properties of the graphs may also give an insight into the meaning of the clusters being formed.

6.3 Other applications

It may be worthwhile to apply the Rauzy graph technique to other (related) areas in biology. Protein molecules within cells are important for various functions of the cell. Proteins are responsible for – amongst other things - regulating reactions in a cell, and regulating the production of specific substances (Campbell et al. 1999). Proteins are polymers, much like DNA, however are comprised of amino acids as opposed to nucleotides. Protein sequences can be represented by a sequence of characters representing the amino acids. It would be interesting if protein sequences could be analysed using Rauzy graphs, although there will be a whole new set of associated challenges, for instance there are 20 possible amino acids as opposed to 4 nucleotide bases.

6.4 Algorithm extension

Finally, it is possible that this technique can be improved by using a modified algorithm for building graphs. This would be achieved by developing a graph algorithm to take into consideration the nature of the type of sequence the graph is being built from (DNA, RNA, amino acids etc.).

For example, a possible algorithm for graphs from DNA sequences might build three graphs simultaneously. Each graph would represent two reading frames (one forwards, one backwards) and only every third node would be added to each graph. In this case it would make sense to work in multiples of three

bases, so they represent complete codons. By using this method, a graph would better represent a coding region.

A different approach might be to treat sub-sequences the same if they code for the same amino acid(s) (assuming the DNA sequence comes from a coding region). This would mean that wobble-bases in codons are accounted for and the graph is a better representation of the protein it codes for.

There are many possible extensions to the Rauzy graph algorithm to incorporate some higher level biological knowledge into the process of generating graphs.

6.5 Classifying DNA

Evidence collected in the course of this study suggests that the information extracted from the Rauzy graphs could be used in a machine learning algorithm. One appropriate machine learning technique might be an instance-based learning algorithm. An appropriate dataset would be used to 'train' the learner and to classify other DNA sequences using the knowledge that the learner has extracted from the datasets (Witten & Frank 2000). This dataset would consist of many (hundreds or more) previously-classified DNA sequences and the relevant properties of their Rauzy graphs. This extension would have the potential to successfully classify DNA sequences.

6.6 Overview

By undertaking some or all of the further work described in this section, the evidence collected in this study suggests that classifying DNA sequences using the properties obtained from their Rauzy graph representations is possible. Doing so could open up new opportunities for learning and research within the field.

7 References

Berstel, J 2002, 'Recent results on extensions of Sturmian words', *International Journal of Algebra and Computation*, vol. 12, no. 1-2, pp. 371-85.

Campbell, N, Reece, J & Mitchell, L 1999, *Biology*, Fifth Edition edn, Benjamin/Cummings.

Cassaigne, J 1995, 'Special factors of sequences with linear subword complexity', *Developments in Language Theory*.

D'Antonio, L 2003, 'Incorporating bioinformatics in an algorithms course', in *Proceedings of the 8th annual conference on innovation and technology in computer science education*, ACM Press, Thessaloniki, Greece, pp. 211-4.

de Bruijn, NG 1946, 'A combinatorial problem', *Nederl. Akad. Wetesch. Proc.*, pp. 758 - 64.

Frid, AE 2001, 'On factor graphs of DOL words', *Discrete Applied Mathematics*, vol. 114, no. 1-3, pp. 121-30.

Gentleman, RC, Carey, VJ, Bates, DM, Bolstad, B, Dettling, M, Dudoit, S, Ellis, B, Gautier, L, Ge, Y, Gentry, J, Hornik, K, Hothorn, T, Huber, W, Iacus, S, Irizarry, R, Leisch, F, Li, C, Maechler, M, Rossini, AJ, Sawitzki, G, Smith, C, Smyth, G, Tierney, L, Yang, JYH & Zhang, J 2004, 'Bioconductor: Open software development for computational biology and bioinformatics', *Genome Biology*, vol. 5, p. R80.

Jaeger, S, Lima, R & Mosse, B 2003, 'Symbolic analysis of finite words: the complexity function', *Bulletin of the Brazilian Mathematical Society*, vol. 34, no. 3, pp. 457-77.

Li, SY, Pearl, DK & Doss, H 2000, 'Phylogenetic tree construction using Markov chain Monte Carlo', *Journal of the American Statistical Association*, vol. 95, no. 450, pp. 493-508.

Lothaire, M 2002, 'Finite and Infinite Words', in M Lothaire (ed.), *Combinatorics on Words*, Cambridge University Press, p. 256.

Mount, D 2001, 'Phylogenetic Prediction', in *Bioinformatics: Sequence and Genome analysis*, Cold Spring Harbour Laboratory Press, Cold Spring Harbour, New York., pp. 237-80.

National Center for Biotechnology Information *FASTA format description*, viewed 3rd August 2005, <http://www.ncbi.nlm.nih.gov/blast/fasta.shtml>.

National Human Genome Research Institute Accessed: 12/10/2005, *Deoxyribonucleic Acid (DNA)*, <http://www.genome.gov/Pages/Hyperion//DIR/VIP/Glossary/Illustration/base_pair.shtml>.

Preiss, BR 1997, *Data Structures and Algorithms with Object-Oriented Design Patterns in C++*.

R Development Core Team 2005, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 3-900051-07-0, <http://www.R-project.org>.

Saitou, N & Nei, M 1987, 'The Neighbor-Joining Method - a New Method for Reconstructing Phylogenetic Trees', *Molecular Biology and Evolution*, vol. 4, no. 4, pp. 406-25.

Saitou, N & Imanishi, T 1989, 'Relative Efficiencies of the Fitch-Margoliash, Maximum-Parsimony, Maximum-Likelihood, Minimum-Evolution, and Neighbor-Joining Methods of Phylogenetic Tree Construction in Obtaining the Correct Tree', *Molecular Biology and Evolution*, vol. 6, no. 5, pp. 514-25.

Sanderson, MJ & Shaffer, HB 2002, 'Troubleshooting molecular phylogenetic analyses', *Annual Review of Ecology and Systematics*, vol. 33, pp. 49-72.

Towell, G & Shavlik, J 1994, 'Refining Symbolic Knowledge Using Neural Networks', paper presented to International Workshop on Multistrategy Learning, New Brunswick, N.J., July 10 - 13 1994.

Uren, P 2005, File Operation Code in Java to D Nahodil.

Watson, J & Crick, F 1953, 'Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid.' *Nature*, vol. 171, p. 737.

Witten, IH & Frank, E 2000, *Data Mining*, The Morgan Kaufmann Series in Data Managment Systems, Morgan Kaufmann.

8 Appendices

A. Chloroplast DNA sequences

See accompanying CD.

B. Animal DNA sequence

```
>epsilon3|gi3367727|R.norvegicus epsilon 3 globin gene.
GAATTCATAATACATAGGACATAATTCTATCCTAGCTGCGATAGTGCATGTGGATGTTACTGC
ATGGAAATATCACCTGTCACTTCAGAGTTAGGTTACAAACACACAAAAAGTGAAAACAGGCTG
GTTAAATATACCTGAAAGATTGGTCACTCAGCATTATTAATATTTAACCAGTCATGAGCATAG
AGAAGGCTAAATTCACACCTTTTTCACCAAGCGGATACACTTTTCCTGGGGGCTGCCAATTGC
AGGGTGACAGCACAGTTCATTCACGGGGTGGATCATGGAAACATGTACATAGAGTAAATTGCT
CTGTTTTCATGTATGAGAGAAGTAAATGTGGAGACATGCTGTATTTCAACACAGTAAGTATTA
TTTACTCACAAACTGTAATGCATTAGGAATTGCTTTTCTGGGGCTGTGGATACTAGGAAGATC
ATAGCCTAAAACTACAGAAAATGCATTCAATGGTTTGCACACCATTTCTTTGTCATTAGTGGC
TTGTAAGTAACAGGCATATTTCTGTCACTTATGTCCTAAGTCACAGGCCTACAGA


>18S|gi76563834|Bos taurus 18S ribosomal RNA gene (540 bases)
TTGTCTCAAAGATTAAGCCATGCATGTCTAAGTACGCACGGCCGGTACAGTGAAACTGCGAAT
GGCTCATTAAATCAGTTATGGTTCCTTTGGTCGCTCGCTCCTCTCCTACTTGGATAACTGTGG
TAATTCTAGAGCTAATACATGCCGACGGGCGCTGACCCCCTTCGCGGGGGGGATGCGTGCATT
TATCAGATCAAAACCAACCCGGTCAGCCTCCTCCCGGCCCCGGCCGGGGGGCGGGCGCCGGCG
GCTTTGGTGACTCTAGATAACCTCGGGCCGATCGCACGCCCCCCGTGGCGGCGACGACCCATT
CGAACGTCTGCCCTATTAACTTTCGATGGTAGTCGCTGTGCCTACCATGGTGACCACGGGTGA
CGGGGAATCAGGGTTCGATTCCGGAGAGGGAGCCTGAGAAACGGCTACCACATCCAAGGAAGG
CAGCAGGCGCGCAAATTACCCACTCCCGACCCGGGGAGGTAGTGACGAAAAATAACAATACAG
GACTCTTTCGGGGCCCTGTAATTGGAATGAGTCCAC


>HomoSapienHBZ|gi183794|Homo sapiens hemoglobin zeta (HBZ) gene
(first 525 bases)
CTCATGAGGCTGAGGCAGAAGAATCACTTGAACCAGGGAGTCAGAGGTTGCAGTGAGCTGGGA
TCGCACCACTGCACTCCACCCTGGGCGACAAATCGAGATTCCATCTCAAAAAAAGAAAAAAAA
ATTAAAAGGAATATTTGCCTCATTATGTTACAATAACTAATATGGAAAGCAATATTGCAATGC
CTATTAGCACATGACATTAGGTGAATTCTCCTTTGTCCCCGGACCTGCTGCCTCCTCCTGCTT
GTCAGGGGACAGATCCAGTACATCTCCCCTCAGCGCTGGGTGGACCTAACCCTTGCTTTCTTG
GAGGAAACCCAGGAATCCAGAGACAAAGTGGAAGGGTACTGGCATGTGGTTGGGCAGGGCTGC
CTGAGGTCGGTGTCAGCCGACCGTGGGGCTTGGTCCCAGGAGGCTGCTTACTGGGCCCTGCTC
CTCTGGTTTCCCCCAAGTCGTGATTCTGAAATGAATAAGGACGGTGCAGAACTGGACTACAAA
TGCAGGAGTGACTTCCTGGGA
```

C. Random DNA sequences

```
>randomSequence1
GGGTTTGGAGAGTAGGAAATGCTGCAATGCATTATTCTGAACCATTCCATGGCTACTACAGGG
GACGCATTCCACAAATGGTCAGTCTGAGACTTGACTACCCCAATTGTCCATTCGGAACCTGAC
```

```
ATACCACAGAACGGAAGGACTAAAATTTAAGAAAGCGGAATTCATCTCCGGATACCTCTCTCC
TGTATGGGCCGCTGCACCAACAGGGAGATTCATCTGACACTCAGGATCTGTCTAGAAACTGAC
GTGCCGATATTAAATAAGGTTTGTCCCTGTCGGGAAAACGAGTTTCACGCCTTGTCGGGTTCA
TTGAACTGATTATTGGCGAGTAACGATACAAGCACGAAAGACGCAACTGACACAACGGCGCTC
ATAAAGGGAGTGGCAACACGACGCCTTGTATCAACCACAAATCACTAGGTCGAGCCCTGATAA
CAGGTAACGGGGCCCAGATACAAGTGCCAGGGTGCCCTCAAGGGACGTTTAGGTAGCAGACCC
CGGTGAGCTTCGTTGAGAATGGGGGAGCAAGTC


>randomSequence2
CGGACAAAGGTCCTTCCTGCTCACGTGATCCTTCAATCCTTCAGAGAAATTGGTTTTGATCGT
GGTAGTTTGAGAACACGGGTTATGCTTGAGTATAATTCCTCGGTTCCGGCCATTGTTCGCAGC
GGGTGTATAGTTAATGCTTCATGTGCAACGGTGTCCCAGCACGGGGTGCTACACGTCGATAGG
CGAGGCTCTAGTTCACGTAGTGCCGTCAGTACGCCTTTAAGGGTCGGAGGAGACAAAGGCGAC
AGGGTATTGATACCAGCCGTGGAACGGGTACCTAAGCAATTGGTTCAGACAGGAAGTATATGA
AATAAAGCCTGTGCCCGCGAAGTGTTACCATCCCTCGCCGATCAGACGTAATTCTGTGGTTGA
ATGGATCCCCGAGGAAATGTGCGCTACTCCTCCCTCGGTCCTCAATTTGCCGGTTACACAATG
CGGCAATCTAGTTTGAGTGGAGGTACCCAACGCGACACT


>randomSequence3
AGGGTAAAATTTGGAGAAGTGAAAAGGTAACCCCGACTGGCTTCGCTCCCCGCTCCGAGAAAT
GGAGGCGAAGACCGCCATTAGACCCTAGGGATAAAATGAGTTACACATGGGGCTTAGGAGGGG
AAACGGGGTCGCGGCTCCGTCTCTTATGTCCGGTCAACGTTCGTGATGTGAACCATTGGTAGG
CCCTCCGGGTTACAATTAACATGCCACAAGCATTAGTTCAGGATGGACATTTTGAGCCTGGAG
TATTTAGACGCTGCACAGACCCGTTGATCTGGTCGTCACGACATACTTATTTCCTCAGGCGTG
CGTGTCCCTTCAGTTATAACTGCCTGTACGATTAATTTCAGATCTATGTCATGTTTGACCCGC
CACCGACCTACGAGACTGGGTTGCAAATCTGTACTAGGGATGTTTGCTGAACTAGTAATTGCT
CCGCGTGTCATACCCCCCGGTATCAGGGGGGCTCGTGACCGCACCGTGGTGTGAGACCCCGGG
CTTGCGTGT


>randomSequence4
CAACAGGGGATGTTGGTCAATTCAAGCTTCGGCCCTGGCGGCGGGTCCAACCGACTCGGCAGA
CTGTAGGTTGCGTACCGGGAACCTAAGGTTTGCCGATATATATTCGCTTCGCGCTGATGTCCT
GGCCTGTCTCCCATACGAAGCCTGTGTACTCCTTACACCGATCTCCACTTGCGACTAGTCTAG
CGGTTCCGGATCAGCTTATTGTATCCACATGATCGTTCGAATATGTACACCAAAAGCTGGCCA
ATAGGTGTAATCCCTATAACGCAGCAGCATATGACAGTGTCCTCCTGATCCAATTCCAGCTCC
AGCCAGACGCTCTAATCGGTGTACTACGTCTCGATAGCGGCTCGAATCTTACCTAATGCCTGA
TTGTTGGCGTCAGTCGAAGTGCATCAGTGGCCTCCTCTACCATCCATAAGGGTTGACGATATC
TCCACCAAAACGGCTCACTTTCTTCGCATAGTGCTGTTGGCAGAGAAACCGGGTTAGAGTCCT
AACGGTAATAGTCTCTTCCCAGATCCGCCCTTCTGGTGAACTATCCTAGACACGCGTGGCAT


>randomSequence5
AATCTCAACGGACACCGAGAACCAGGTCTTAAGCTAGGGTGGCTGTCCAAATTTGCACTTAGG
TTCCCTGAATGCAAAACCGGAATCTTTAGGACGCCGACGATGTTTAGTCAAATCAGCCGGAGT
ATCCGGTCAGTAATCGCAACATTAGGACGGAGGCGTCTCCTTGGTACGTAGTATCGCGTCCCC
AGCTTCACGCCAGAGGAATGGCGCCGGCCAGCACCTTCAATAGAGGAGGAAAACCGACGTTAG
AGCTGTTATGCTACTGTCAAGCATTGTGGTGCCGGGTAGCGTAGCAACCTACCGGAGACTGAT
CACGACCCGCTCATCTTCCTTAAAAAAAGTTAGCATAGTGGCTTTTGTCAGTGGCCTTTATGG
GTACTTACTTAAGTCGAGCTTAGAAGGTTTTCGCCTGACATCTCAGTTGACGGGGCCACGCCT
GTGTAGCGGCCAAAAGGCGGCTCTAAGCTAGAGCTTAAGGAGAAGAGTAGAAGTCCTAGAGAA
TTT
```

D. Summaries of data collected (orders 2 – 10)

See accompanying CD.